

# The Singleton Pattern



**Gerald Britton**

Pluralsight Author

@GeraldBritton [www.linkedin.com/in/geraldbritton](http://www.linkedin.com/in/geraldbritton)



## Overview



**Classification: Creational**

**Ensure a class has only one instance**

**Control access to limited resource**

- Device access
- Buffer pools
- Web/DB connection pools

**Provide a global point of access**

**Class responsible for its one instance**

**Lazy construction**



## Demo

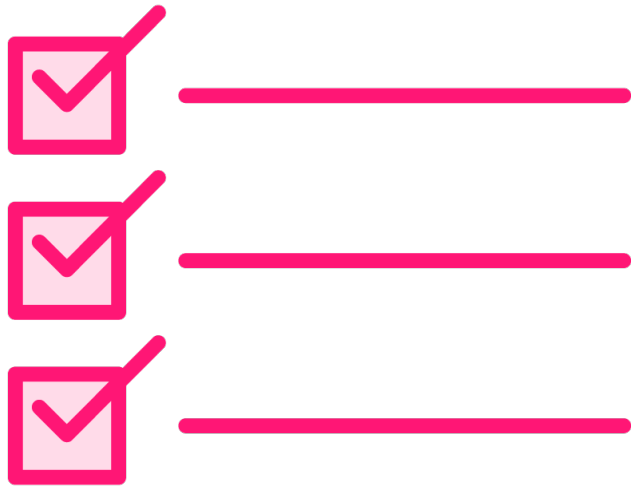


## Motivating example

### Logging subsystem:

- Log events to a file
- Only one instance can write to the file
- Need to control access
- Classic Singleton pattern





**Violate Single Responsibility Principle**

**Non-standard class access**

**Harder to test**

**Carry global state**

**Hard to sub-class**

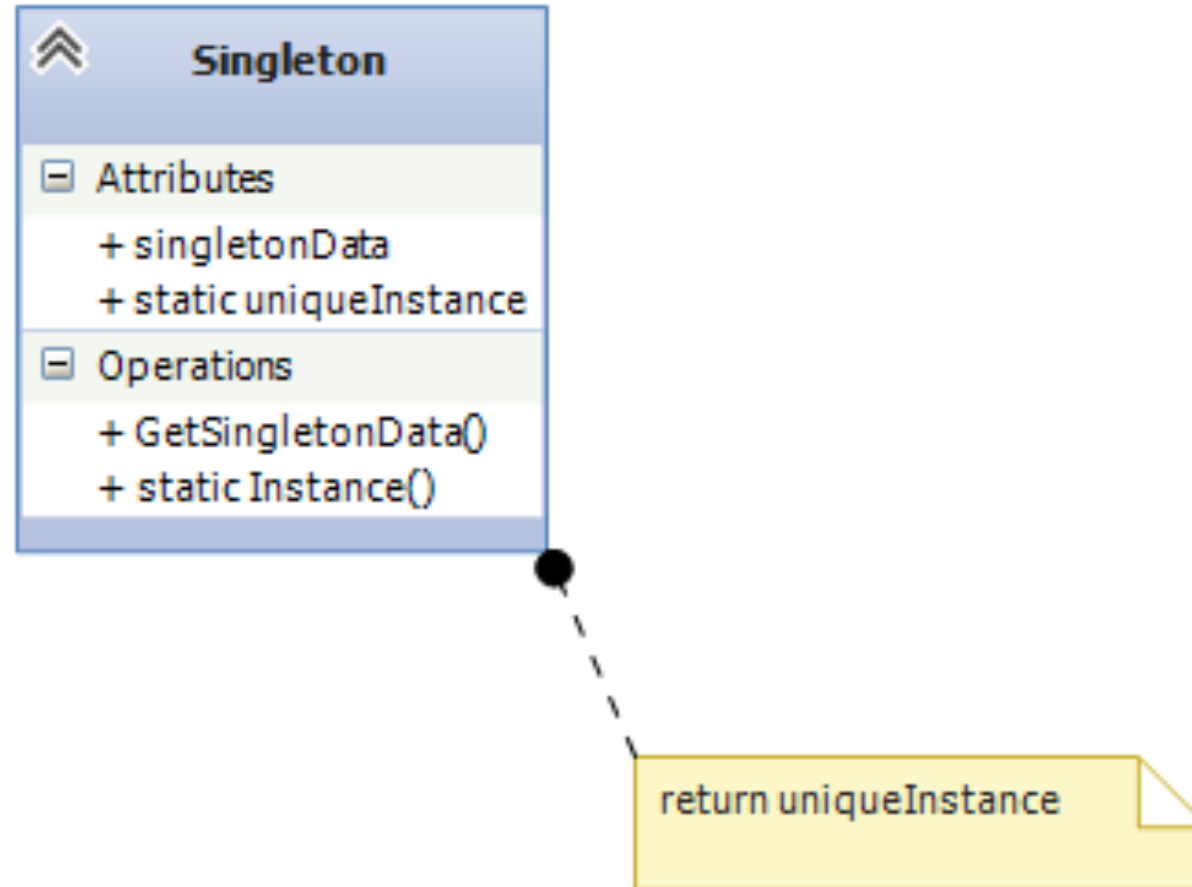
**Singletons considered harmful!**

- <http://goo.gl/VUWmC6>
- <http://goo.gl/O4s3VE>

**Singletons called an anti-pattern**



# Singleton Pattern Structure



## Demo



### Fix the Single Responsibility problem

- Building a base class for all singletons
- Inherit from the base class for each one
- Fix non-standard instance access
- Other problems remain



## Demo



### First demo, classic pattern

- Single Responsibility violation

### Second demo, built a base class

- Fixed the SRP violation

### Third demo, build a metaclass

- Class's class
- Class is an instance of a metaclass
- Control building of class





## Demo



**What!? Another demo?**

**First demo, classic pattern**

**Second demo, built a base class**

**Third demo, used a metaclass**

**Fourth demo, the MonoState pattern**





## Summary



**Controlled access to a single instance**

**Reduces the global namespace**

**Subclassible for extended uses**

**Variable number of instances**

- Base class and metaclass variants

**More flexible than a static class**

- Class with no instances

**MonoState shares all state**

**Can also use a Python module**

**Use sparingly – *anti-pattern***

